EXHIBIT H

← All posts

Hello Taro: Building the (Tap)Root of the World's Financial Network with Bitcoin 👋🍠

Michael Levin
September 28, 2022

Today we're excited to announce the alpha release of the Taro daemon, enabling developers to mint, send, and receive assets on the bitcoin blockchain. In April, we first announced Taro, a Taproot-powered protocol for issuing assets that can be transferred over bitcoin and in the future, the Lightning Network for instant, high volume, low fee transactions. We are grateful to the bitcoin developer community for their valuable feedback, and have incorporated it into the draft Bitcoin Improvement Proposals (BIPs), which specify the protocol, and the Taro alpha daemon implementation.

In the coming months, more enhanced features will be added to the Taro daemon including universe functionality. Universes will allow users and asset issuers to provide proofs about asset provenance, supply issuance, and more easily interact with Taro asset data. Once the on-chain functionality is complete, we'll work towards integrating the Taro protocol into `lnd`, bringing Taro assets to the Lightning Network. The initial work to implement simple, unannounced Taproot channels in `lnd` has begun, a prerequisite for Lightning channels that can send and receive Taro assets. We're incredibly excited to get the new Lightning capabilities into the developer community's hands!

To get started with the Taro alpha daemon, check out the readme on the repo, download the daemon, review the API documentation, and read the getting started guide. Note that this initial release is only designed for testnet usage as a way for developers to start using the code. For a more extensive explanation on how Taro works, take a deep dive into the BIPs and our documentation.

We're releasing this initial version of the daemon to continue to solicit feedback from the community and build this open source protocol in public. Given the alpha, testnet-only nature of the daemon, we encourage developers to explore how Taro will fit into their products, understanding that it will continue to be revised and improved as we progress to a mainnet release.

The First Step Towards Bitcoinizing the Dollar

Before outlining the daemon setup, let's revisit the "why" behind Taro. The release marks the first step towards bitcoinizing the dollar by 1) issuing assets, like stablecoins, on the most decentralized and secure blockchain, bitcoin and 2) allowing users to transact those assets on the most performant and efficient global payments network, Lightning.

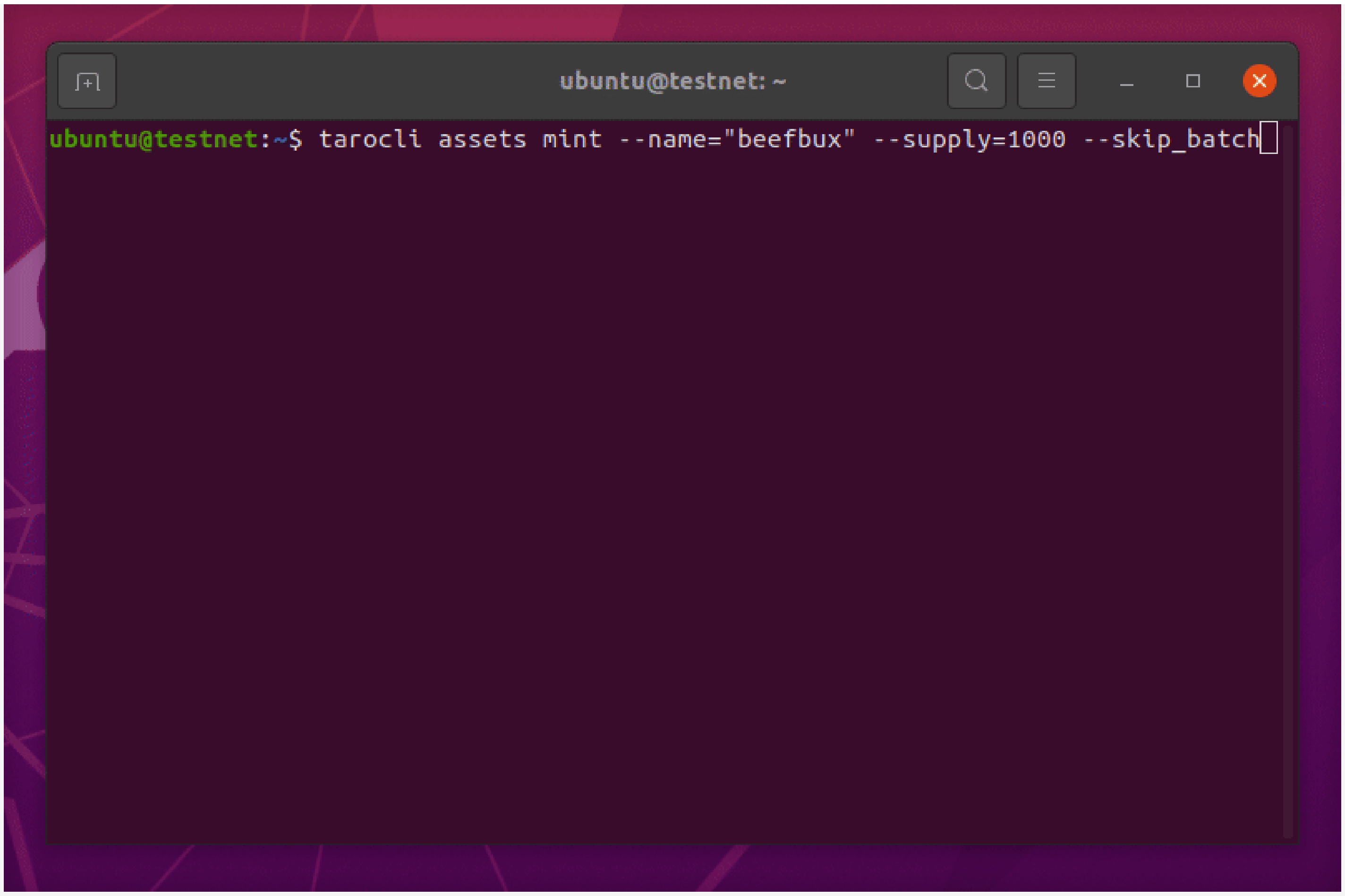
In talking to bitcoin and Lightning developers across the world, we've heard that users want to use stablecoins in the same way they're using bitcoin on the Lightning Network: instantly settled, low-fee, peer-to-peer transactions without financial intermediaries. Taro will enable applications around the world like Strike, Ibex Mercado, Paxful, Breez, and Bitnob to give their users access to bitcoin- and Lightning-native stablecoins.

With Taro and the incredible developer community, we can build a world where users have USD-denominated balances and BTC-denominated balances (or other assets) in the same wallet, trivially sending value across the Lightning Network just as they do today. This leap forward will accelerate the path to bringing bitcoin to billions.

Taro Mint, Send, and Receive: Oh My!

As a refresher, Taro assets are embedded within existing bitcoin outputs, or UTXOs. Think of these assets as "UTXOs within a UTXO." A developer mints a new Taro asset by making an on-chain transaction that commits to special metadata in a Tapoot output. When minting a new asset, the Taro daemon will generate the relevant witness data, assign the asset to a private key held by the minter, and broadcast the newly created bitcoin UTXO to the bitcoin network. This new outpoint becomes the genesis point of the newly minted asset, acting as its unique identifier. Taro minting has a few key design attributes: verifiability, ability to issue fungible assets like currencies, and scalability.

To see these design attributes in action, we have created a proof of concept gif showing the minting of 1000 "beefbux" on testnet, in homage to our CTO Laolu Osuntokun (pseudonym roasbeef).



```
ubuntu@testnet:~$ tarocli assets mint --name="beefbux" --supply=1000 --skip_batch
```

Verifiability is a core design attribute of the Taro minting process as it is vital that users receiving Taro assets can easily verify that the received assets were minted by the expected issuer. This verifiability attribute is accomplished with the creation of a globally-unique asset ID upon minting. This asset ID includes the `asset_genesis` field, as seen in the above gif, which describes the provenance, or the origination point of a Taro asset by specifying the genesis point (i.e. the specific on-chain outpoint in which the asset was first issued). This component of the minting process ensures receivers can verify that the received coins can be traced back to the expected issuance event.

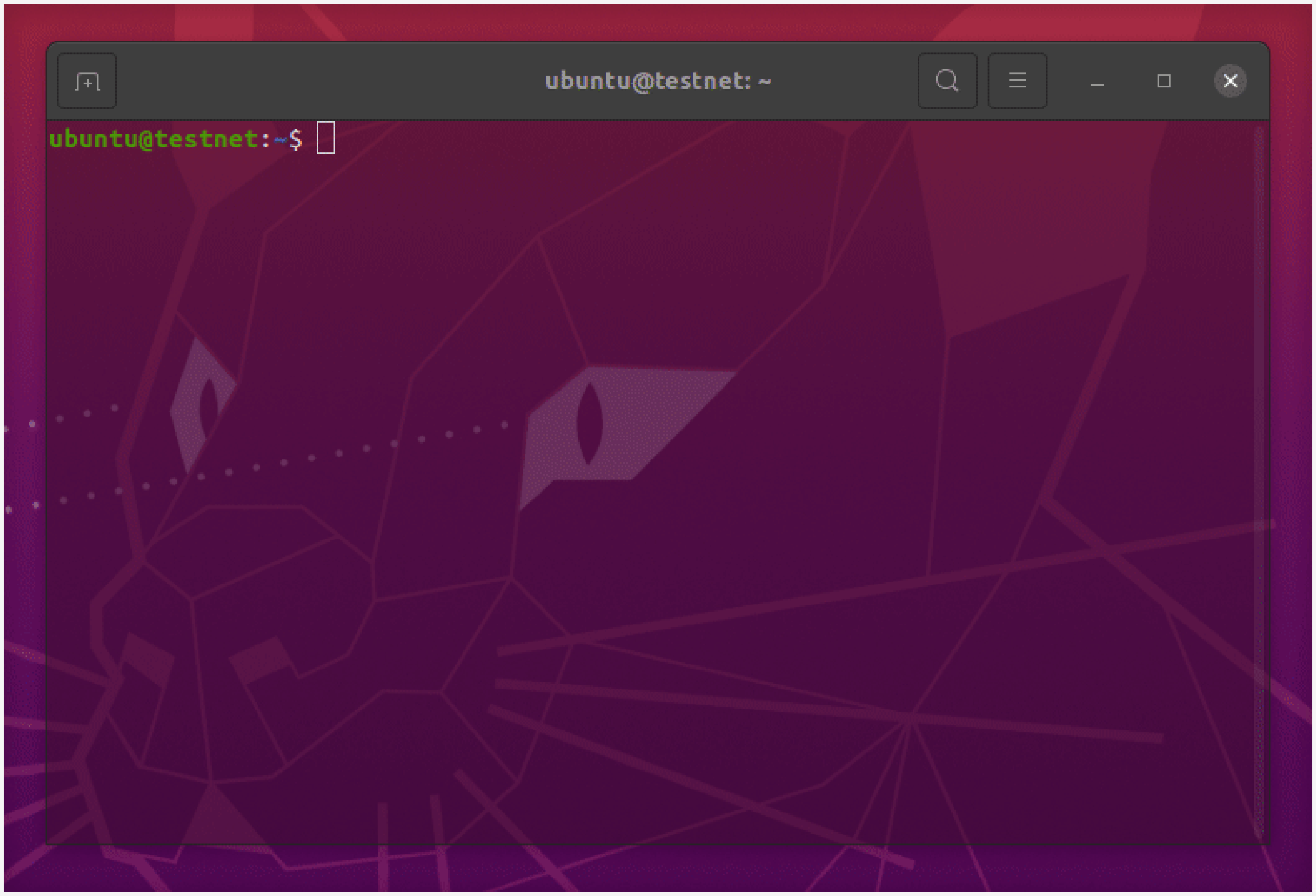
Assets like stablecoins are typically issued in repeatable minting events ("tranches"), but need to retain interchangeability -- or fungibility -- across tranches. In other words, a 5 dollar bill printed in 1985 needs to be exchangeable for a 5 dollar bill printed in 2022. The same needs to be true for Taro stablecoin assets. This fungibility property of an asset is encoded with the parameter `asset_type`, which is set as `NORMAL` (as in the gif). Further, minters can link different sets of the same asset under the same family key to achieve the properties necessary for interchangeability for the same assets minted in different tranches.

Taro's on-chain efficiency and scalability is critically important; hence the design enables multiple Taro actions to be completed in a single on-chain transaction. In the demo above, a single tranche of Taro issuance is created, but it is possible today to perform multiple mints in the same bitcoin transaction. In the future, it will also be possible to perform multiple Taro actions (minting, sending, etc.) in a single bitcoin transaction. Combining Taro transactions into one bitcoin transaction is accomplished with the parameter `chain_anchor`. The `anchor_txid` parameter describes the transaction in which the new assets were issued, which will be visible both from a user's local node and from any public block explorer. This property is particularly useful for asset issuers given the cost of minting a batch can be amortized over a single on-chain output.

Next, the Taro address format, used during sending and receiving, has some novel properties and oft-requested features to deliver smooth UX:

tarot1qqqsqq3qtv7gupgd4lec2gpzpkcywm33823lqlh0a8utxuz80sa2a2a68p5sgg9q4l43vhcwcd5gpd5wpw4tmxkeccharf564xvtcv8f5drzqts83uyq8lwv4qhv5ul6

The Taro address BIP specifies in detail how implementations can generate a Taro address, and the information it encodes. One notable feature is that the parameters identifying which asset is expected to be received are encoded **within the address**, making it impossible for a sender to accidentally send the wrong asset to a receiver. No more horror stories about users accidentally sending assets to an exchange that did not support said asset -- with Taro that's no longer a concern!



```
ubuntu@testnet:~$ 
```

Towards Taro on Lightning

Taro enables both bitcoin and Lightning to be multi-asset networks. Users will be able to open Taro channels that plug in at the edges and interoperate with the existing Lightning Network. Instead of starting from scratch and bootstrapping a new ecosystem of nodes and liquidity, Taro will leverage the existing network effects of both the infrastructure that's been built out over the last several years plus the nearly 5,000 BTC allocated to the network today as a global routing currency. The Lightning Network and

Taro are uniquely positioned to serve this market need given a pre-existing broad and deep payment channel graph.

For routing nodes in the core of the network, Taro's "edge node" model should mean more volume with no upgrades required as Taro transactions route through BTC. For users at the edges of the network, it should mean all the technological benefits of the Lightning Network -- instant settlement, efficient fees, global reach -- with access to their preferred Unit of Account.

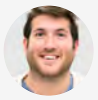
In order to bring Taro to the Lightning Network, simple, unannounced Taproot channels will go through the process of initial spec review as well as cross-implementation compatibility tests to then be merged into `lnd`. This ongoing work is the main feature of `lnd 0.16`. Once completed and launched, we'll move into the next phase of development for enabling Taro assets on the Lightning network.

Say Hello to Taro today!

To get started exploring Taro, download the daemon, check out the API documentation, and read the getting started guide. And for a more extensive explanation on how Taro works, take a deep dive into the Taro BIPs and our documentation.

We're incredibly excited to get this open source alpha daemon code into the hands of the developer community to build with, and hope this progress gets people excited for what's to come. If you're interested in helping us develop Taro directly, we are actively hiring protocol engineers and infrastructure devs. If you haven't already, join the Taro Channel in our Slack Community, reach out to us on Twitter, attend the lnd PR review club, contribute a PR, or subscribe to our newsletter!

Next article →



About the author
Michael Levin

Michael received his Bachelor of Science in Economics from the Wharton School at the University of Pennsylvania with a double concentration in Management and Operations, Information, and Decisions along with a minor in Engineering Entrepreneurship. Before Lightning, Michael worked across a variety of teams and functions at Google including product, growth, and marketing. Michael loves food, travel, live music, skiing, and sports.

Keep in touch

Get the latest news

The Lightning Lab

Lightning Network updates, community coverage, and, of course, memes!

Type your email... Subscribe

substack

Contact us

@Lightning

Developer Slack Signup

Lightning Network Daemon

hello@lightning.engineering

LIGHTNING LABS

Products

Team

Join Us

Blog